#include <GL/glut.h>

#include<string.h>

#define PLAYER\_ONE 0

#define PLAYER\_TWO 1

#define NO\_WINNER\_TILL\_NOW (-1)

void mainPage ();

int page=0;

// Points to draw board.

GLdouble boardPoints [][3] = {

{-100.0, -100.0, 10.0},

{100.0, -100.0, 10.0},

{100.0, -100.0, -10.0},

{-100.0, -100.0, -10.0},

{100.0, 100.0, 10.0},

{100.0, 100.0, -10.0},

{-100.0, 100.0, -10.0},

{-100.0, 100.0, 10.0}

};

// Points to draw lines on the board.

GLdouble lines [][3] = {

{-33.33, -100.0, 11.0},

{-33.33, 100.0, 11.0},

{33.33, -100.0, 11.0},

{33.33, 100.0, 11.0},

{-100.0, 33.33, 11.0},

{100.0, 33.33, 11.0},

{-100.0, -33.33, 11.0},

{100.0, -33.33, 11.0}

};

// Points to display heading.

GLdouble headingPoints [][3] = {

{-100.0, 100.0, 10.0},

{100.0, 100.0, 10.0},

{100.0, 100.0, -10.0},

{-100.0, 100.0, -10.0},

{100.0, 120.0, 10.0},

{100.0, 120.0, -10.0},

{-100.0, 120.0, -10.0},

{-100.0, 120.0, 10.0}

};

double boxPoints [4][3] = {

{-100, 100, 12},

{-33.33, 100, 12},

{-33.33, 33.33, 12},

{-100, 33.33, 12}

};

/\*\* 2D array to store progress of game.

\* -1 => Box is empty.

\* 0 => Cone is marked.

\* 1 => Cube is marked. \* /

int gameProgress [3][3] = {

{-1, -1, -1},

{-1, -1, -1},

{-1, -1, -1}

};

/\*

\* Variable to keep track of turn.

\* Even => Player 1.

\* Odd => Player 2.

\*/

int turn = 0;

// Player who won the game.

int winner = NO\_WINNER\_TILL\_NOW;

// Variables to identify which box is selected.

int row = 0;

int column = 0;

// Main window ID.

int mainWindow;

// Game window ID.

int gameWindow;

// Winner window ID.

int winnerWindow;

// Credits window ID.

int creditsWindow;

// Variable to indicate match drawn.

int draw = -1;

// Mouse function for home page.

void mainMouse (int button, int state, int x, int y);

// Keyboard function for home page.

void mainKeyboard (unsigned char key, int x, int y);

// Function to get input from arrow keys

void arrowKeyInput (int key, int xMouse, int yMouse);

// Function to open home page

void openMainWindow ();

void resetGame () {

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

gameProgress[i][j] = -1;

}

}

turn = 0;

winner = NO\_WINNER\_TILL\_NOW;

row = 0;

column = 0;

draw = -1;

}

void polygon (GLdouble (\*points) [3], int a, int b, int c, int d) {

glBegin (GL\_POLYGON);

glVertex3dv(points[a]);

glVertex3dv(points[b]);

glVertex3dv(points[c]);

glVertex3dv(points[d]);

glEnd ();

}

// Function to write text.

void writeStrokeText (char \*string, float x, float y, float z) {

glPushMatrix ();

glTranslatef (x, y, z + 1);

glScalef (0.09f, -0.08f, z);

glRotatef (180, 1, 0, 0);

glEnable (GL\_LINE\_SMOOTH);

for (char \*c = string; \*c != '\0'; c++) {

glutStrokeCharacter (GLUT\_STROKE\_MONO\_ROMAN, \*c);

}

glPopMatrix ();

}

void writeNormalText (char \*string, float x, float y) {

glRasterPos2f (x, y);

for (char \*c = string; \*c != '\0'; c++) {

glutBitmapCharacter (GLUT\_BITMAP\_HELVETICA\_18, \*c);

}

}

void moveBox () {

if (winner == NO\_WINNER\_TILL\_NOW && turn < 9) {

// Move row.

switch (row) {

case 1:

glTranslated (0.0, -66.66, 0.0);

break;

case 2:

glTranslated (0.0, -66.66 \* 2, 0.0);

break;

default:

break;

}

// Move column.

switch (column) {

case 1:

glTranslated(66.66, 0.0, 0.0);

break;

case 2:

glTranslated(66.66 \* 2, 0.0, 0.0);

break;

default:

break;

}

}

}

void drawCone(int row, int column) {

glPushMatrix();

glTranslatef((GLfloat) ((column - 1) \* 66.66), (GLfloat) ((row - 1) \* -66.66), 10);

glScalef(10, 10, 50);

glutSolidCone(1, 1, 36, 1);

glPopMatrix();

}

void drawCube(int row, int column) {

glPushMatrix();

glTranslatef((GLfloat) ((column - 1) \* 66.66), (GLfloat) ((row - 1) \* -66.66), 35);

glScalef(10, 10, 25);

glutSolidCube(2);

glPopMatrix();

}

// Function to select the box on the board.

void drawBoxOnSelected() {

glColor3f(1, 0, 0);

if (winner == NO\_WINNER\_TILL\_NOW && turn < 9) {

glLineWidth(7);

glPushMatrix();

moveBox();

glBegin(GL\_LINE\_LOOP);

glVertex3dv(boxPoints[0]);

glVertex3dv(boxPoints[1]);

glVertex3dv(boxPoints[2]);

glVertex3dv(boxPoints[3]);

glEnd();

glPopMatrix();

}

}

void drawBoard() {

// Board.

glColor3f(0, 0, 1);

polygon(boardPoints, 0, 1, 2, 3);

glColor3f(0, 1, 0);

polygon(boardPoints, 0, 1, 4, 7);

glColor3f(1, 0, 0);

polygon(boardPoints, 1, 2, 5, 4);

glColor3f(1, 0, 1);

polygon(boardPoints, 4, 5, 6, 7);

glColor3f(0, 1, 1);

polygon(boardPoints, 2, 3, 6, 5);

glColor3f(1, 1, 0);

polygon(boardPoints, 0, 3, 6, 7);

// Heading.

glColor3f(0, 0, 1);

polygon(headingPoints, 0, 1, 2, 3);

glColor3f(0.71, 0.57, 0.45);

polygon(headingPoints, 0, 1, 4, 7);

glColor3f(1, 0, 0);

polygon(headingPoints, 1, 2, 5, 4);

glColor3f(1, 0, 1);

polygon(headingPoints, 4, 5, 6, 7);

glColor3f(0, 1, 1);

polygon(headingPoints, 2, 3, 6, 5);

glColor3f(1, 1, 0);

polygon(headingPoints, 0, 3, 6, 7);

glColor3f(0, 0, 0);

// Border for heading.

glLineWidth(7);

glBegin(GL\_LINE\_LOOP);

glVertex3i(-100, 100, 10);

glVertex3i(100, 100, 10);

glVertex3i(100, 120, 10);

glVertex3i(-100, 120, 10);

glEnd();

// Border for board.

glBegin(GL\_LINE\_LOOP);

glVertex3i(-100, 100, 10);

glVertex3i(-100, -100, 10);

glVertex3i(100, -100, 10);

glVertex3i(100, 100, 10);

glEnd();

// Draw lines on board.

glLineWidth(5);

for (int i = 0; i < 8; i += 2) {

glBegin(GL\_LINES);

glVertex3dv(lines[i]);

glVertex3dv(lines[i + 1]);

glEnd();

}

// Heading

writeStrokeText("TIC - TAC - TOE", -60, 105, 10);

drawBoxOnSelected();

// Indicate whose turn to play.

if (turn < 9 && winner == NO\_WINNER\_TILL\_NOW) {

if (turn % 2)

drawCube(row, column);

else

drawCone(row, column);

}

// Fill boxes which are already marked.

for (int row = 0; row < 3; row++) {

for (int column = 0; column < 3; column++) {

if (gameProgress[row][column] == PLAYER\_ONE) {

drawCone(row, column);

} else if (gameProgress[row][column] == PLAYER\_TWO) {

drawCube(row, column);

}

}

}

}

int checkForWinner() {

// Check row

for (int i = 0; i < 3; i++) {

if (gameProgress[i][0] == gameProgress[i][1]

&& gameProgress[i][0] == gameProgress[i][2]

&& gameProgress[i][0] != -1) {

return gameProgress[i][0];

}

}

// Check column

for (int i = 0; i < 3; i++) {

if (gameProgress[0][i] == gameProgress[1][i]

&& gameProgress[0][i] == gameProgress[2][i]

&& gameProgress[0][i] != -1) {

return gameProgress[0][i];

}

}

// Check diagonal

if (gameProgress[0][0] == gameProgress[1][1]

&& gameProgress[0][0] == gameProgress[2][2]

&& gameProgress[0][0] != -1) {

return gameProgress[0][0];

}

// Check diagonal

if (gameProgress[0][2] == gameProgress[1][1]

&& gameProgress[0][2] == gameProgress[2][0]

&& gameProgress[0][2] != -1) {

return gameProgress[0][2];

}

// Check for draw

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

if (gameProgress[i][j] == -1)

return -1;

}

}

draw = 1;

return -10;

}

// Display function for game window

void gameDisplay() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

drawBoard();

glutSwapBuffers();

}

void gameInit() {

glClearColor(0.2, 0.6, 0.8, 0);

glOrtho(-320, 320, -180, 180, -250, 250);

glRotatef(-40, 1, 0, 0);

}

void winnerInit() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glOrtho(0, 640, 0, 360, -50, 50);

glutSwapBuffers();

}

// Display function to display winner

void playerWinDisplay() {

glColor3f(0.0, 0.0, 0.8);

glLineWidth(6);

glBegin(GL\_LINE\_LOOP);

glVertex2i(153, 277);

glVertex2i(153, 83);

glVertex2i(487, 83);

glVertex2i(487, 277);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2i(160, 270);

glVertex2i(160, 90);

glVertex2i(480, 90);

glVertex2i(480, 270);

glEnd();

glColor3f(0.6, 0.3, 0.0);

if (winner == PLAYER\_ONE) {

writeNormalText("Player One Won!!!", 280, 180);

} else if (winner == PLAYER\_TWO) {

writeNormalText("Player Two Won!!!", 280, 180);

} else if (draw == 1) {

writeNormalText("Match Draw!!!", 280, 180);

}

glColor3f(0, 0, 0.85);

// Button to go to home page

glBegin(GL\_LINE\_LOOP);

glVertex2i(180, 64);

glVertex2i(460, 64);

glVertex2i(460, 12);

glVertex2i(180, 12);

glEnd();

glColor3f(1, 0, 0);

writeNormalText("Go to homepage", 280, 30);

glutSwapBuffers();

}

void playerWinKeyboard(unsigned char key, int x, int y) {

switch (key) {

case 'X':

case 'x':

glutDestroyWindow(winnerWindow);

openMainWindow();

break;

default:

break;

}

}

void mainDisplay() {

glClearColor(0, 0, 0, 0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(0, 0, 0.85);

if(page==0)

{

mainPage();

glLoadIdentity();

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 640, 0, 360);

//glFlush();

}

if(page==1){

// First button

glBegin(GL\_LINE\_LOOP);

glVertex2i(160, 288);

glVertex2i(480, 288);

glVertex2i(480, 216);

glVertex2i(160, 216);

glEnd();

// Second button

glBegin(GL\_LINE\_LOOP);

glVertex2i(160, 144);

glVertex2i(480, 144);

glVertex2i(480, 72);

glVertex2i(160, 72);

glEnd();

glColor3f(1, 0, 0);

writeStrokeText("Instruction", 275, 108, 0);

writeStrokeText("Start Game", 275, 252, 0);

writeStrokeText("TIC - TAC - TOE", 250, 325, 0);

}

glutSwapBuffers();

}

void playerWinMouse(int button, int state, int x, int y) {

float newX = (float) (640.0 / 1366) \* x;

float newY = (float) (360.0 / 768) \* y;

if (newX < 480 && newX > 160 && (360 - newY) < 64 && (360 - newY) > 12) {

mainWindow = glutCreateWindow("Tic Tac Toe - 3D");

glutDestroyWindow(winnerWindow);

glutFullScreen();

gluOrtho2D(0, 640, 0, 360);

glutDisplayFunc(mainDisplay);

glutKeyboardFunc(mainKeyboard);

glutMouseFunc(mainMouse);

}

}

void openMainWindow() {

mainWindow = glutCreateWindow("Tic Tac Toe - 3D");

glutFullScreen();

gluOrtho2D(0, 640, 0, 360);

glutDisplayFunc(mainDisplay);

glutKeyboardFunc(mainKeyboard);

glutMouseFunc(mainMouse);

}

void gameKeyboard(unsigned char key, int x, int y) {

switch (key) {

// Rotate in x axis

case 'S':

case 's':

glRotatef(5, 1, 0, 0);

break;

case 'W':

case 'w':

glRotatef(-5, 1, 0, 0);

break;

// Rotate in y axis

case 'D':

case 'd':

glRotatef(5, 0, 1, 0);

break;

case 'A':

case 'a':

glRotatef(-5, 0, 1, 0);

break;

// Rotate in z axis

case 'Q':

case 'q':

glRotatef(5, 0, 0, 1);

break;

case 'E':

case 'e':

glRotatef(-5, 0, 0, 1);

break;

case ' ':

if (turn <= 9 && gameProgress[row][column] == -1 && winner == NO\_WINNER\_TILL\_NOW) {

if ((turn % 2) == PLAYER\_ONE) {

drawCone(row, column);

} else {

drawCube(row, column);

}

if (gameProgress[row][column] == -1) {

gameProgress[row][column] = (turn % 2);

winner = checkForWinner();

// Display winner

if (winner != NO\_WINNER\_TILL\_NOW) {

winnerWindow = glutCreateWindow("Game Over!");

glutSetWindow(winnerWindow);

glutDestroyWindow(gameWindow);

glutFullScreen();

winnerInit();

glutDisplayFunc(playerWinDisplay);

glutKeyboardFunc(playerWinKeyboard);

glutMouseFunc(playerWinMouse);

}

}

turn++;

}

break;

case 'X':

case 'x':

glutDestroyWindow(gameWindow);

openMainWindow();

break;

default:

break;

}

glutPostRedisplay();

}

float height = 0;

void creditsDisplay() {

glClearColor(0, 0, 0, 0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(1, 1, 1);

writeNormalText("Instruction", 210, height - 20);

writeNormalText("start game --> right click on mouse", 210, height - 40);

writeNormalText("move pawn up, down, left, right --> 'use four arrow keys' ", 210, height - 60);

writeNormalText("place pawn on distict position --> 'space key' ", 210, height - 80);

writeNormalText("rotate game board x-axis --> 'W/w and S/s keys'",210, height - 100);

writeNormalText("rotate game board y-axis --> 'D/d and A/a keys'", 210, height - 120);

writeNormalText("rotate game board z-axis --> 'Q/q and E/e keys'", 210, height - 140);

writeNormalText("quit window or game --> 'X/x key'", 210, height - 160);

if (height <= 220) {

height += 1;

}

glutPostRedisplay();

glutSwapBuffers();

}

void creditsKeyboard(unsigned char key, int x, int y) {

switch (key) {

case 'X':

case 'x':

// Reset height to 0, to get scrolling animation again when it's opened.

height = 0;

glutDestroyWindow(creditsWindow);

openMainWindow();

break;

default:

break;

}

}

void mainMouse(int button, int state, int x, int y) {

float newX = (float) (640.0 / 1366) \* x;

float newY = (float) (360.0 / 768) \* y;

if (newX < 480 && newX > 160 && (360 - newY) < 288 && (360 - newY) > 216) {

// Start game button

gameWindow = glutCreateWindow("Tic Tac Toe - 3D");

glutDestroyWindow(mainWindow);

glutFullScreen();

resetGame();

glutDisplayFunc(gameDisplay);

gameInit();

glutKeyboardFunc(gameKeyboard);

glutSpecialFunc(arrowKeyInput);

glEnable(GL\_DEPTH\_TEST);

} else if (newX < 480 && newX > 160 && (360 - newY) < 144 && (360 - newY) > 72) {

// Credits button

creditsWindow = glutCreateWindow("Credits");

glutDestroyWindow(mainWindow);

glutFullScreen();

gluOrtho2D(0, 640, 0, 360);

glutDisplayFunc(creditsDisplay);

glutKeyboardFunc(creditsKeyboard);

}

}

void mainKeyboard(unsigned char key, int x, int y) {

switch (key) {

case 'X':

case 'x':

glutDestroyWindow(mainWindow);

case 'p':

page=1;

mainDisplay();

break;

default:

break;

}

}

// Function to get input from arrow keys

void arrowKeyInput(int key, int xMouse, int yMouse) {

switch (key) {

case GLUT\_KEY\_UP:

if (row > 0)

row -= 1;

break;

case GLUT\_KEY\_DOWN:

if (row < 2)

row += 1;

break;

case GLUT\_KEY\_LEFT:

if (column > 0)

column -= 1;

break;

case GLUT\_KEY\_RIGHT:

if (column < 2)

column += 1;

break;

default:

break;

}

drawBoxOnSelected();

glutPostRedisplay();

}

void print(int x,int y, unsigned char \*string, void \*font)

{

int len,k;

glRasterPos2f(x,y);

len=strlen(string);

for (int i = 0; i < len; i++) {

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, string[i]);}

}

void mainPage()

{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT); //clear the window

glColor3f(0,1,0); //text color's.

print(100,330,"SAI VIDYA INSTITUTE OF TECHNOLOGY ,Rajanukunte, Bengaluru- 560 064",GLUT\_BITMAP\_TIMES\_ROMAN\_24); //display

glColor3f(1,0,1); //text color's.

print(1,310, " (Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka ",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

print(6,300, "| Approved by AICTE, New Delhi) ",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,1);

print(150,270,"DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,0,0);

print(150,200,"COMPUTER GRAPHICS - TIK-TAK-TOE GAME SIMULATION",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0.7,0,1);

print(350,70,"Under the guidance of :",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,1);

print(300,50," Dr.Sangeetha.V - ",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,1);

print(300,40," Sunil G L - ",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,1);

print(420,50,"Associate Professor, Dept. of ISE",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,1);

print(380,40,"Assistant Professor, Dept. of CSE",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,0);

print(10,50,"Harshita T - 1VA17CS015",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(1,1,0);

print(10,40,"Jagbeer Poonia - 1VA17CS018",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0.5,1,0.1);

print(20,150,"(Press p to start)",GLUT\_BITMAP\_TIMES\_ROMAN\_24);

}

void main(int argc, char \*\*argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

mainWindow = glutCreateWindow("Tic Tac Toe - 3D");

glutFullScreen();

gluOrtho2D(0, 640, 0, 360);

glutDisplayFunc(mainDisplay);

glutKeyboardFunc(mainKeyboard);

glutMouseFunc(mainMouse);

glutMainLoop();

}